JSP (Java Sever Page)

- JSTL

01. JSTL의 개념과 구성

기본적인 프로그래밍 요소

1. JSTL이란?

- JSTL(JSP Standard Tag Library)은 커스텀 태그 라이브러리 기술을 이용해서 일반적으로 필요한 기능들을 표준화한 것으로 크게 핵심(CORE), xml, I18N(국제화), 데이터베이스(SQL), 함수(functions) 라이브러리로 구성된다.
- 커스텀 태그 기반이므로 JSTL을 사용하는 방법은 일반적인 커스텀 태그와 같다.
- 다운로드 : http://tomcat.apache.org/taglibs/standard/
- taglib 지시어 사용법(core 라이브러리의 경우)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

라이브러리	URI	prefix
핵심	http://java.sun.com/jsp/jstl/core	c
XML	http://java.sun.com/jsp/jstl/xml	x
I18N	http://java.sun.com/jsp/jstl/fmt	fmt
데이터베이스	http://java.sun.com/jsp/jstl/sql	sql
함수	http://java.sun.com/jsp/jstl/functions	fn

01. JSTL의 개념과 구성

2. JSTL 구성

- JSTL은 태그가 제공하는 기능의 성격에 따라 5가지 주요 라이브러리로 구분된다.

라이브러리	기능	태그	접두어
핵심	General Purpose Actions	• catc • out • remove • set	c
	Conditional Actions	• choose • when • otherwise • if	
	Iterator Actions	• forEach • forTokens	
	URL Related Actions	• import • redirect • url • param	
XML	Core	• out • parse • set	x
	Flow Control	• choose • when • otherwise • forEach • if	
	Transformation	• transform • param	

출력
변수삭제
변수선언 (속성에저장)
switch case 문 과 비슷
반복문
문자열을 잘라준다
url : 경로를 만들어준다

01. JSTL의 개념과 구성

– I18N 은 다국어 처리와 관련된 기능을 제공하고 sql(데이터베이스)는 간단하게 데이터베이스 관련 작업을 지원하며 functions(함수)는 여러 부가기능들을 제공한다.

라이브러리	기능	태그	접두어
I18N	Locale	• setLocale	fmt
	Message Formatting	• bundle • message • param • setBundle • requestEncoding	
	Number and DateFormatting	• formatNumber • formatDate • parseDate • parseNumber • setTimeZone • timeZone	
데이터베이스	Database Access	• setDataSource • query • dateParam • param • transaction • update • dateParam • param	sql
함수		contains • ontains • containsIgnoreCase • endsWith • escapeXml • indexOf • join • length • replace • split • startsWithsubstring • substringAfter • substringBefore • toLowerCase • toUpperCase • trim	fn

■ 01. JSTL의 개념과 구성

3. JSTL 학습 사전 준비

- JSTL은 규격화된 커스텀 태그 라이브러리를 배우는 것으로 특별히 어려운 점은 없으나 각 태그의 기능을 이해하고 활용하려면 적절한 데이터가 미리 준비되어 있어야 한다.
- 여기서는 톰캣 시작 시 InitialMember 리스너 클래스에서 Member 클래스 객체 10개를 생성하고 샘플 데이터로 초기화하는 작업을 자동으로 수행한다.

- JSTL 설치와 리스너 클래스 복사

tomcat.apache.org/taglibs/standard/에서 1.1.2 버전을 다운로드 한 후, 압축을 풀고 [lib] 폴더에서 [illegible].jar 파일을 찾아 프로젝트의 [WEB-INF₩lib] 폴더로 복사.

02. 핵심 라이브러리의 주요 태그

1. 기본 기능 태그

- **<c:out> 태그**
  - 초기에는 jsp 표현식(<%= %>)을 대체하기 위해 개발 되었으나 표현언어가 JSP에 기본으로 제공 되면서 사용 빈도는 줄었으나 몇몇 옵션은 유용하게 사용할 수 있다.
    - 태그 바디가 없는 경우

```
<c:out value="value" [escapeXml="{true|false}"] [default=defaultValue"]/>
```

    - 태그 바디가 있는 경우 (출력값)

```
<c:out value="value" [escapeXml="{true|false}"]>
default value (value에 내용이 없을 때 출력될 기본 값)
</c:out>
```

속성	필수	기본 값	설명
value	Y	없음	출력될 내용 또는 표현식이다.
default	N	태그 바디에 있는 내용	value 값에 내용이 없는 경우에 출력할 내용으로, 태그 바디 혹은 속성 값 형태로 올 수 있다.
excapeXml	N	true	출력될 내용에 〈 〉, &, ', " 등의 문자를 일반 문자로 변환할 것인지 결정한다. 예를 들어 출력될 내용에 HTML 태그가 포함되어 있다면 이 값을 false로 해야 태그가 반영된 내용이 화면에 보인다. 만일 true로 할 경우 태그가 그대로 화면에 보이게 된다.

02. 핵심 라이브러리의 주요 태그

- out.jsp
  - 리스너에 등록된 members 객체(ArrayList)의 내용을 출력.
  - <c:forEach>는 for 문의 역할을 하는 JSTL로, 뒤에 다시 살펴봄.
  - 이름과 이메일을 출력하고 이메일이 없을때는 “email 없음” 이라고 빨간색으로 출력함.

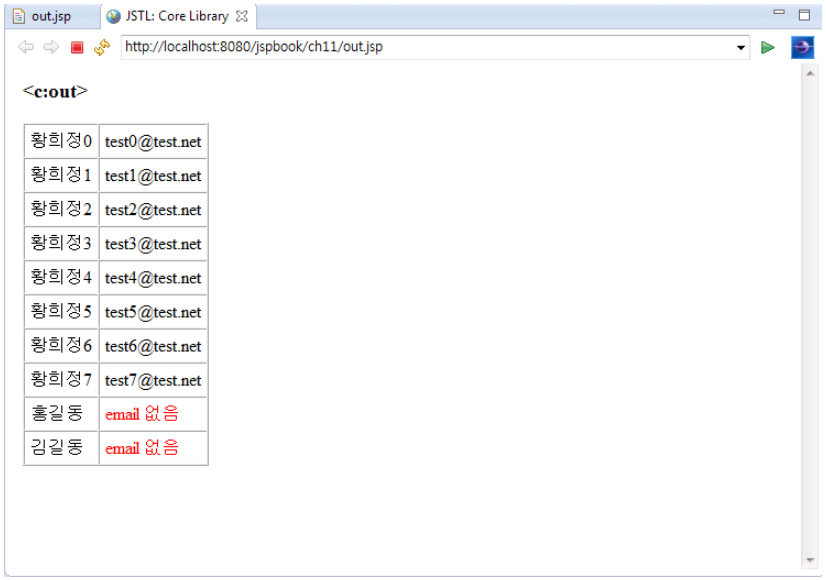
```
<table border="1" cellpadding=5 cellspacing=0>
    <c:forEach var="member" items="${members}">
        <tr>
            <td><c:out value="${member.name}"/></td>
            <td><c:out value="${member.email}" escapeXml="false">
                <font color=red>email 없음</font>
                </c:out>
            </td>
        </tr>
    </c:forEach>
</table>
```

02. 핵심 라이브러리의 주요 태그

- 14행의 <c:out value="${member.name}"/> 은 ${member.name}로 대체할 수 있음.
- JSP 표현식으로 표현할 경우 다음과 같이 됨.

```
<%= ((Member)pageContext.getAttribute("member")).getName() %>
<%= ((Member)application.getAttribute("member")).getName() %>
```

- 15행의 escapeXml="false"는 표현하는 컨텐츠에 HTML 태그가 있을 경우 이를 해석해서 보여줌. true의 경우에는 태그 그대로 출력.



02. 핵심 라이브러리의 주요 태그

- **<c:set> 태그** view에서 많이 사용되는기능이 아님.
  - **<c:set> 태그는 변수 값을 설정하거나 객체의 멤버변수 값을 설정할 때 사용한다.**
  - **사용법**
    - **해당 범위에 속성 값을 추가하는 경우(바디가 올 수도 있다)**

```
<c:set value="value" var="varName" [scope="{page|request|session|application}"]/>
```

    - **특정 target 객체에 새로운 속성 값을 설정하는 경우(바디가 올 수도 있다)**

```
<c:set value="value" target="target" property="propertyName"/>
```

속성	필수	기본 값	설명
value	N	없음	저장할 변수 값
target	N	없음	값이 저장될 객체 이름
property	N	없음	target 객체의 멤버변수 이름
var	N	없음	값이 저장될 변수 이름
scope	N	page	값이 저장될 범위(page, session, request, application)

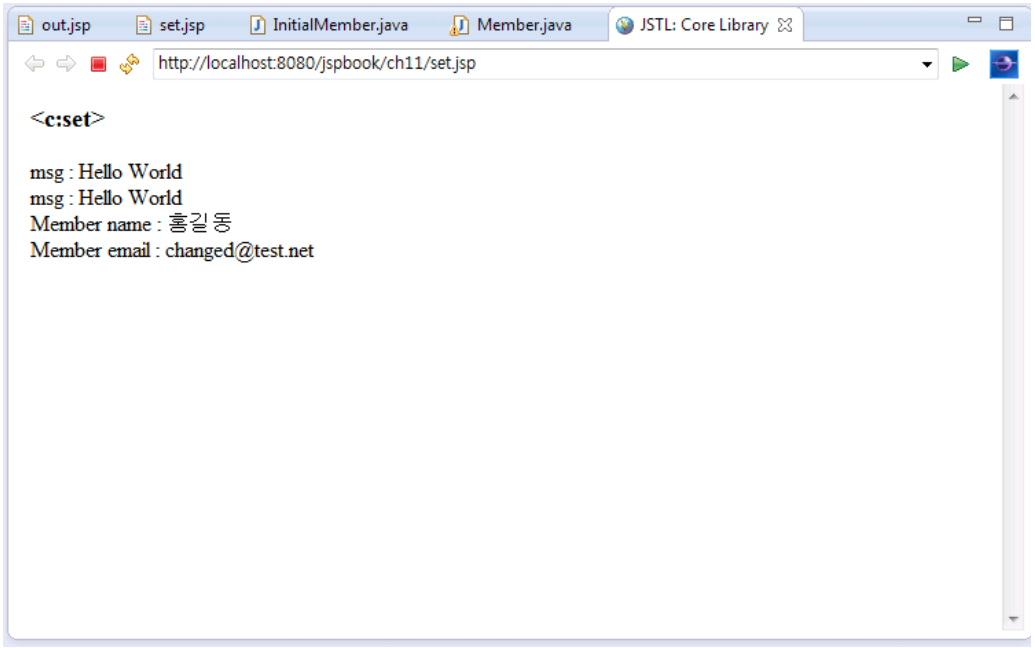
02. 핵심 라이브러리의 주요 태그

- set.jsp
  - 임의 변수에 값을 지정하고 출력함.
  - 리스너에 의해 등록된 객체의 내용을 변경함.

```
09  <h3>&lt;c:set&gt;</h3>
10  <c:set value="Hello World" var="msg"/>
11  msg : ${msg} <BR>
12  msg : <%=pageContext.getAttribute("msg") %><BR>
13
14  <c:set target="${member}" property="email" value="changed@test.net" />
15  Member name : ${member.name} <BR>
16  Member email : ${member.email}
```

  - 12 ~ 14행 : 기존 객체의 멤버변수 값을 설정하는 예로 InitialMember 리스너에서 만든 member 객체의 email 변수를 changed@test.net으로 변경하고 있다.
  - 초기 값은 test@ test.net으로 들어가 있다. 출력해보면 변경된 값이 출력되는 것을 알 수 있다. 이름은 초기 설정 값인 '홍길동'으로 나온다.

02. 핵심 라이브러리의 주요 태그



02. 핵심 라이브러리의 주요 태그

- **<c:remove> 태그**
  - **<c:remove> 태그는 해당 scope 에 설정된 객체를 제거 한다.**
  - **사용법**

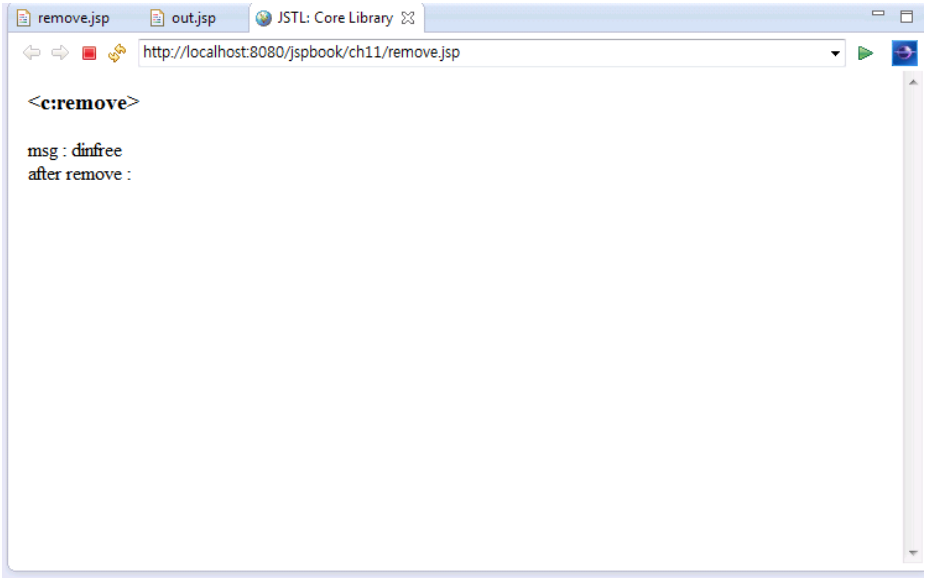
```
<c:remove var="varName" [scope="{page|request|session|application}"]/>
```

속성	필수	기본 값	설명
var	Y	없음	삭제할 변수 이름
scope	N	모든 범위	삭제할 범위

02. 핵심 라이브러리의 주요 태그

- remove.jsp
  - 임의 변수에 값을 설정한 다음 출력하고 삭제하고 다시 출력해 봄.

```
09  <c:set value="Hello World" var="msg" />
10  before remove : ${msg} <BR>
11  <c:remove var="msg" />
12  after remove : ${msg}
```



02. 핵심 라이브러리의 주요 태그

- <c:catch> 태그
  - <c:catch> 태그는 바디에서 실행되는 코드의 예외를 처리한다.
  - JSP를 뷰 역할에 충실하게 프로그래밍 한다면 크게 사용할 일은 많지 않다.
  - 사용법

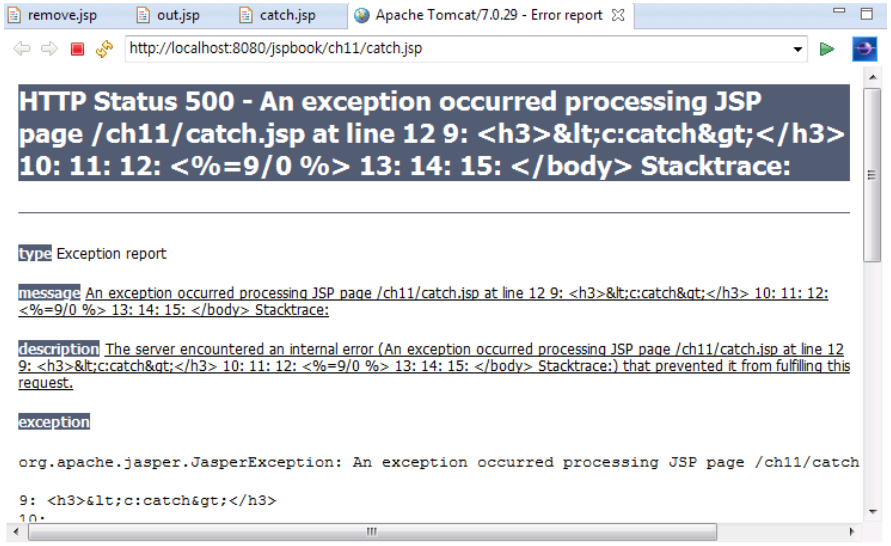
```
<c:catch [var="varName"]>
nested actions
</c:catch>
```

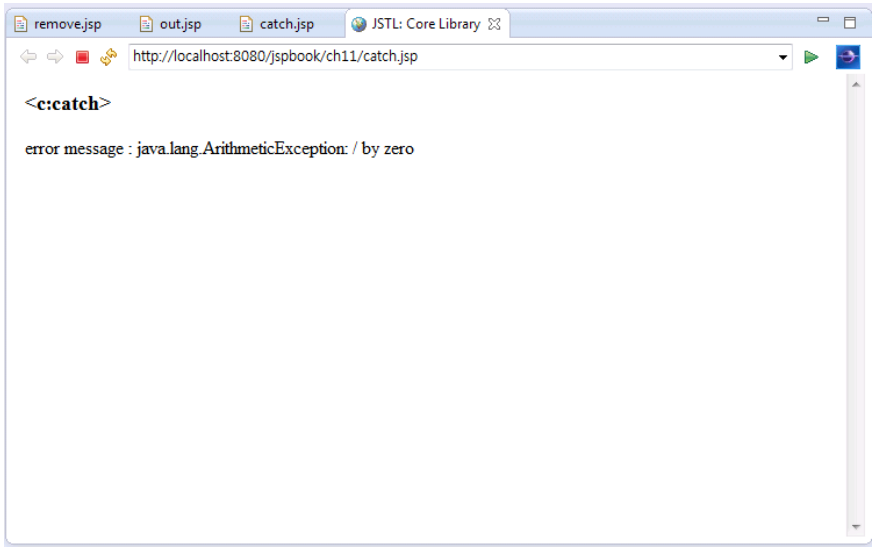
속성	필수	기본 값	설명
var	Y	없음	오류 메시지를 저장할 변수 이름

02. 핵심 라이브러리의 주요 태그

- catch.jsp
  - 스크립트릿을 이용해 예외를 발생시켜 <c:catch> 동작을 확인.

```
09  <c:catch var="errMsg">
10      <%=9/0 %>
11  </c:catch>
```





02. 핵심 라이브러리의 주요 태그

2. 조건 처리 태그

분기해주는 태그

- **<c:if> 태그**
  - <c:if> 태그는 조건에 따라 바디 내용을 처리 한다.
  - 자바의 if와 비슷하지만 else문은 지원하지 않는다(단순 조건 체크만 가능).
  - 사용법
    - 바디 내용이 없는 경우

```
<c:if test="testCondition" var="varName"
[scope="{page|request|session|application}"]/>
```

바디 내용이 있는 경우

EL : 비교연산, 논리연산

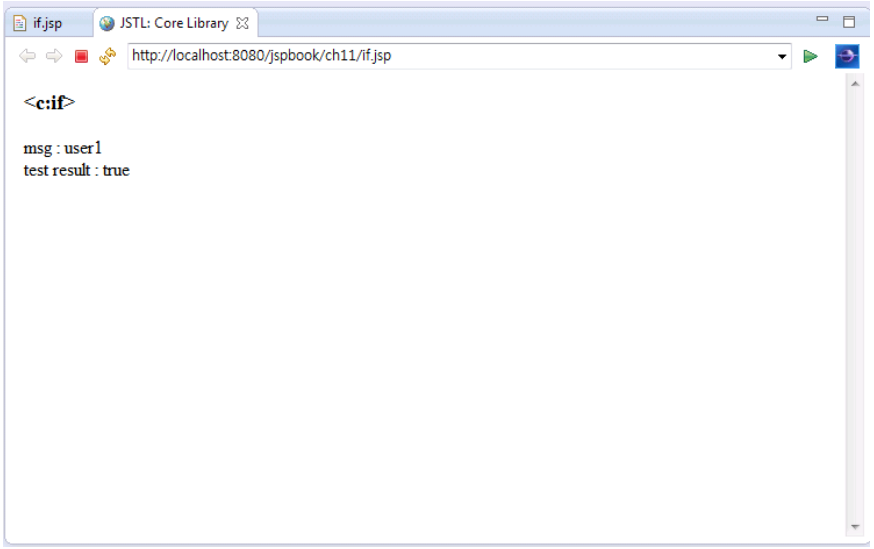
```
<c:if test="testCondition" [var="varName"]
[scope="{page|request|session|application}"]>
Body content
</c:if>
```

속성	필수	기본 값	설명
test	Y	없음	검사할 조건
var	N	없음	test 조건의 결과를 저장할 변수(결과는 true 혹은 false)
scope	N	page	변수가 저장될 범위

02. 핵심 라이브러리의 주요 태그

- if.jsp
  - <c:set>을 이용해 값을 설정 하고 조건 체크를 통해 <c:if> 동작을 확인

```
09  <c:set value="user1" var="msg" />
10     msg : ${msg}" <BR>
11
12  <c:if test="${msg == 'user1'}" var="result">
13    Test result : ${result}
14  </c:if>
```



02. 핵심 라이브러리의 주요 태그

- <c:choose>,<c:when>,<c:otherwise> 태그
  - 이들 태그는 함께 사용되며 자바의 if ~ else if 문, switch 문과 유사하다.
  - <c:choose> 태그 내에는 <c:when> 태그가 여러 개 올 수 있다.

```
<c:choose>
        body content (<when> and <otherwise> subtags)
        <c:when test="testCondition">
                body content
        </c:when>
        <c:otherwise>
                conditional block
        </c:otherwise>
</c:shoose>
```

속성	필수	기본 값	설명
test	Y	없음	검사할 조건

02. 핵심 라이브러리의 주요 태그

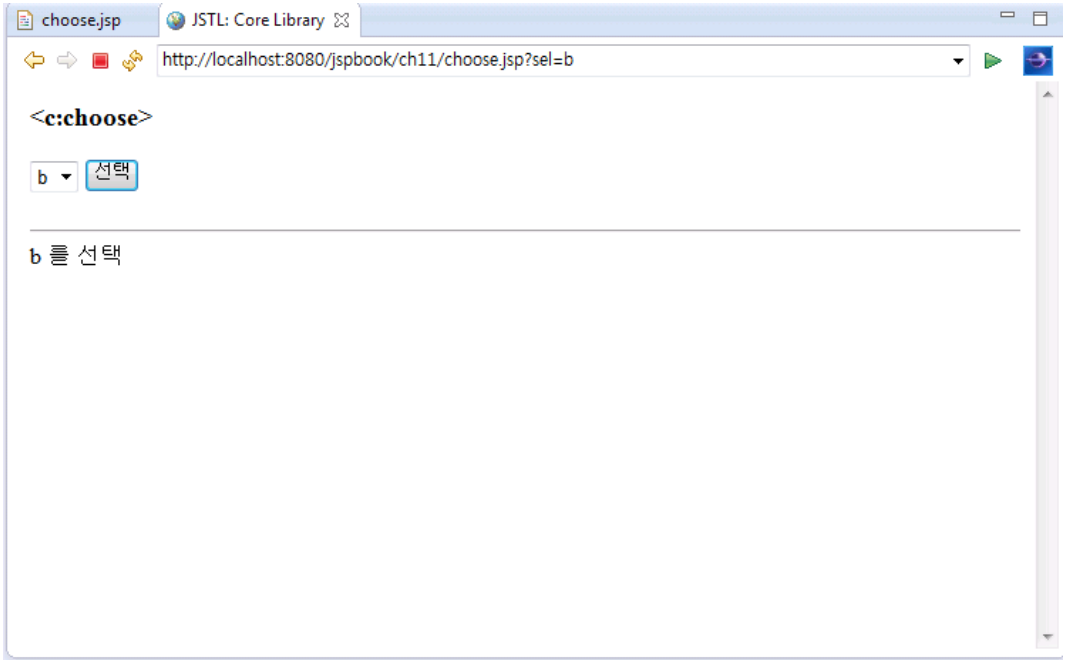
- choose.jsp
  - HTML 폼으로부터 사용자 입력을 받고 선택된 값을 조건문에 의해 처리.

```
21 <c:choose>
22     <c:when test="${param.sel == 'a'}" >
23         a를 선택
24     </c:when>
25     <c:when test="{param.sel == 'b'}" >
26         b를 선택
27     </c:when>
28     <c:when test="${param.sel == 'c'}" >
29         c를 선택
30     </c:when>
31     <c:otherwise>
32         a,b,c 외의 것을 선택
33     </c:otherwise>
34 </c:choose>
```

02. 핵심 라이브러리의 주요 태그

- ${param.sel}은 request.getParameter()와 같은 역할을 하는 표현언어
- 조건에 확인되지 않는 값인 경우 <c:otherwise>에서 처리됨.
- 선택 결과 확인시 <select>가 초기화되지 않고 유지되게 하려면 다음과 같이 표현언어를 활용할 수 있음.

```
<option ${param.sel=='a'? 'selected':''}>a</option>
```



02. 핵심 라이브러리의 주요 태그

if 랑 같이 가장 많이 쓰인다.

3. 순환 처리 태그

- <c:forEach> 태그
  - 반복문과 관련된 태그로 자바의 for 문과 유사하다. 가장 중요하고 널리 쓰이는 JSTL 태그 중 하나임.
  - 여러 옵션 활용법을 잘 익혀 두어야 한다.
    - 컬렉션 객체의 크기만큼 반복

var <- 요소를 담는 변수

```
<c:forEach[var="varName"] items="collection" [varStatus="varStatusName"]
[begin="begin"] [end="end"] [step="step"]>
body content
</c:forEach>
```

생략가능

index값 , count값을 구할수잇음

    - 지정된 횟수 반복

```
<c:forEach [var="varName"] [varStatus="varStatusName"] begin="begin"
end="end"
[step="step"]>
body content
</c:forEach>
```

i++(1씩증가)

02. 핵심 라이브러리의 주요 태그

- varStatus는 반복 과정에서 프로그램적으로 필요한 여러 정보를 제공함.
- 자바 언어에서의 for와 같은 자유도는 없기 때문에 모든 반복 처리에 <c:forEach>를 사용하기는 어렵고 데이터 생성시 <c:forEach>에서 처리되기 편리한 형태로 가공해주는 것이 필요하다.

속성	필수	기본 값	설명
items	N	없음	반복을 위한 데이터를 가진 아이템의 컬렉션
begin	N	0	반복 시작 번호
end	N	컬렉션의 마지막 값	반복 끝 번호
step	N	1	반복의 증가분
var	N	없음	현재 아이템이 있는 변수
varStatus	N	없음	반복 상태 값이 있는 변수

02. 핵심 라이브러리의 주요 태그

- <c:forTokens> 태그
  - <c:forTokens>태그는 기본적으로 for문과 유사한 동작을 하지만 문자열을 토큰으로 구분해 처리하는 기능을 제공한다.
  - 자바의 StringTokenizer 클래스와 유사하다고 볼 수 있다.
  - 토큰은 문자열을 구분하기 위한 캐릭터로 “-”, tab, 공백 등 동일한 규칙으로 문자열을 구성해야 한다.
  - 예를 들어 전화번호는 “010-1234-1234” 와 같이 표현하며 이때 토큰은 “-” 이 된다.

```
<c:forTokens items="stringOfTokens" delims="delimiters"
[var="varName"]
[varStatus="varStatusName"]
[begin="begin"] [end="end"] [step="step"]>
body content
</c:forTokens>
```

02. 핵심 라이브러리의 주요 태그

- 기본적으로 <c:forEach>와 동일하며 delims를 통해 토큰(구분자)를 지정한다.

속성	필수	기본 값	설명
items	N	없음	반복을 위한 데이터를 가진 아이템의 컬렉션
delims	Y	없음	구분자(Delimeter)로 사용할 문자
begin	N	0	반복 시작 번호
end	N	컬렉션의 마지막 값	반복 끝 번호
step	N	1	반복의 증가분
var	N	없음	현재 아이템이 있는 변수
varStatus	N	없음	반복 상태 값이 있는 변수

02. 핵심 라이브러리의 주요 태그

- for.jsp
  - 리스너로 등록된 members 객체를 출력하면서 <c:forEach>의 다양한 옵션을 사용해 봄.
  - “,”를 토큰으로 하는 주소록 데이터를 <c:forTokens>를 이용해 처리함.

```
10  <c:forEach var="i" items="${members}" begin="0" varStatus="status" end="5">
11      index: ${status.index} /
12      count: ${status.count} <BR>
13      name: ${i.name} <BR>
14      email: ${i.name} <BR>
15  <HR>
16  </c:forEach>
17
18  <c:forTokens items="홍길동,011-211-0090,서울" delims="," var="sel">
19      ${sel}<BR>
20  </c:forTokens>
```

02. 핵심 라이브러리의 주요 태그

4. URL 관련 태그

- <c:import> 태그
  - 특정 URL 페이지를 현재 페이지에 포함시킨다.
  - <jsp:include> 액션과 유사하다.
    - 포함하고자 하는 자원을 문자열 형태로 포함하는 경우

```
<c:import url="url" [context="context"] [var="varName"]
[scope="{page|request|session|application}"] [charEncoding="charEncoding"]>
optional body content for <c:param> subtags
</c:import>
```

    - 포함하고자 하는 자원을 Reader 객체로 포함하는 경우

```
<c:import url="url" [context="context"] varReader="varReaderName"
[charEncoding="charEncoding"]>
body content where varReader is consumed by another action
</c:import>
```

02. 핵심 라이브러리의 주요 태그

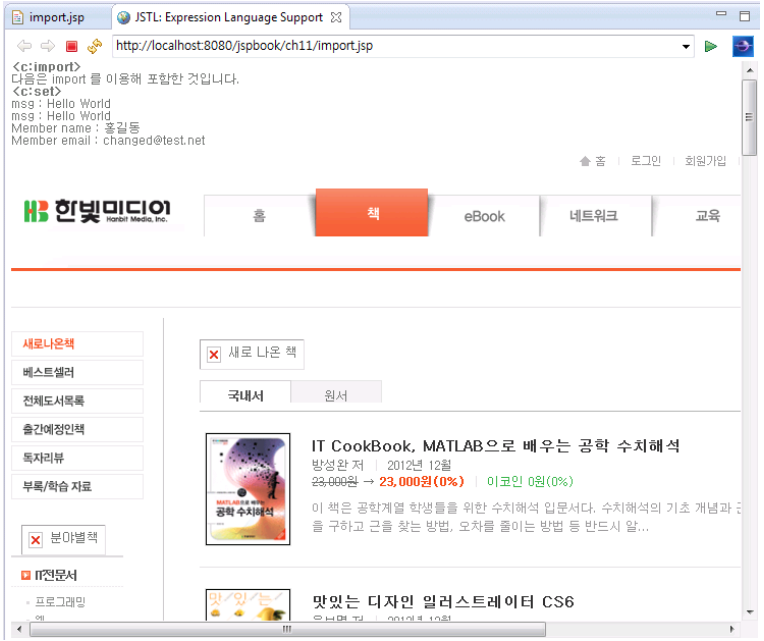
- 동적인 페이지를 포함할 때 사용할수는 있으나 성능상에 문제가 발생할 수 있다.
- 포함한 페이지는 <c:out>을 이용해 출력하기 때문에 용량이 큰 페이지의 사용은 자제하는 것이 좋다.

속성	필수	기본 값	설명
url	Y	없음	현재 페이지 내에 포함시킬 URL
context	N	Current application	현재 웹 애플리케이션 컨텍스트 이름
charEncoding	N	ISO-8859-1	현재 페이지 내에 포함시킬 페이지 캐릭터셋을 지정
var	N	Print to page	포함할 페이지의 내용을 가지는 변수 이름
scope	N	page	var의 범위
varReader	N	없음	자원 내용을 읽으려는 java.io.Reader 변수

02. 핵심 라이브러리의 주요 태그

- import.jsp
  - 동일 서버에 있는 jsp를 포함하거나 외부 url 자원을 포함할 수 있다.
  - escapeXml="false"는 포함될 자원의 HTML 태그를 해석해서 보여준다.

```
12 <c:import url="set.jsp" var="myurl" />
13 <c:out value="${myurl}" escapeXml="false"/>
14 <HR>
15 <c:import url="http://www.hanb.co.kr/book/newbooks.html" var="myurl2" />
16 <c:out value="${myurl2}" escapeXml="false"/>
```



02. 핵심 라이브러리의 주요 태그

- **<c:url> 태그** 경로생성 tag -> contextPath
  - URL Rewriting 즉, 제공된 URL에 파라미터 등을 추가해 프로그램에서 URL 관련 처리(링크 등)를 손 쉽게 할 수 있는 기능을 제공한다.
    - 바디가 없는 경우

```
<c:url value="value" [context="context"] [var="varName"]
[scope="{page|request|session|application}"]/>
```

    - 바디가 있는 경우

```
<c:url value="value" [context="context"] [var="varName"]
[scope="{page|request|session|application}"]>
<c:param> subtags
</c:url>
```

02. 핵심 라이브러리의 주요 태그

- 파라미터는 <c:param> 태그를 이용해 추가할 수 있다.
- scope 지정을 통해 처리하고자 하는 url 객체를 공유할 수 있다.

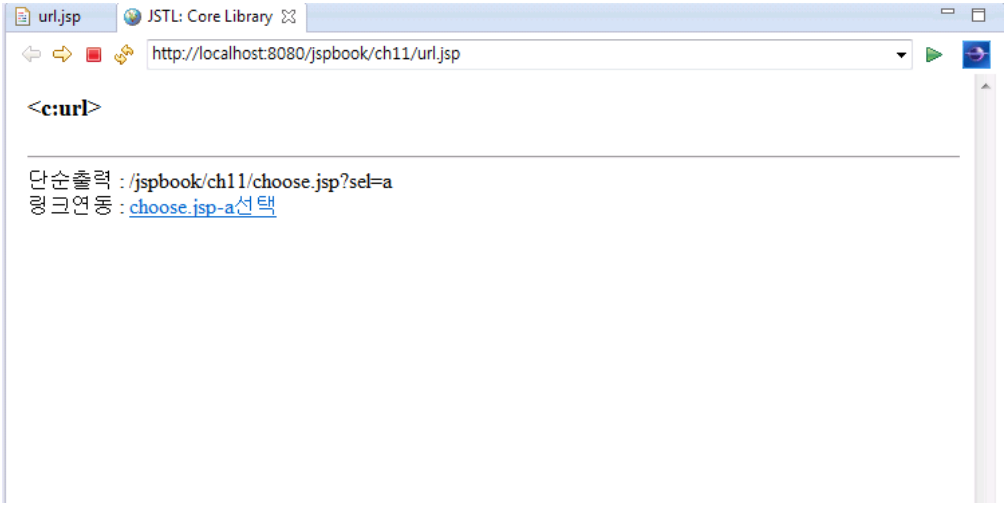
속성	필수	기본 값	설명
value	Y	없음	기본 URL
context	N	Current application	현재 웹 애플리케이션의 컨텍스트 이름
var	N	Print to page	포함할 페이지의 내용을 가지는 변수 이름
scope	N	page	var의 범위

02. 핵심 라이브러리의 주요 태그

- **url.jsp**
  - **url 파라미터를 추가해 본다.**
  - **수정된 url 정보를 HTML에서 표현언어를 이용해 활용한다.**

```
10 <c:url value="/ch11/choose.jsp" var="target">
11     <c:param name="sel">a</c:param>
12 </c:url>
13 <HR>
14 단순 출력: ${target}<BR>
15 링크 연동: <a href="${target}">choose.jsp-a선택</a>
```

choose.jsp?sell=a 가된다.



02. 핵심 라이브러리의 주요 태그

- **<c:redirect> 태그**
  - **response.sendRedirec() 메서드나 <jsp:forward> 액션과 유사하다.**
  - **지정된 페이지로 사용자 요청을 이동시키며 <c:param>을 통해 파라미터 추가도 가능하다.**
    - **바디가 없는 경우**

```
<c:redirect url="value" [context="context"]/>
```

    - **바디가 있는 경우**

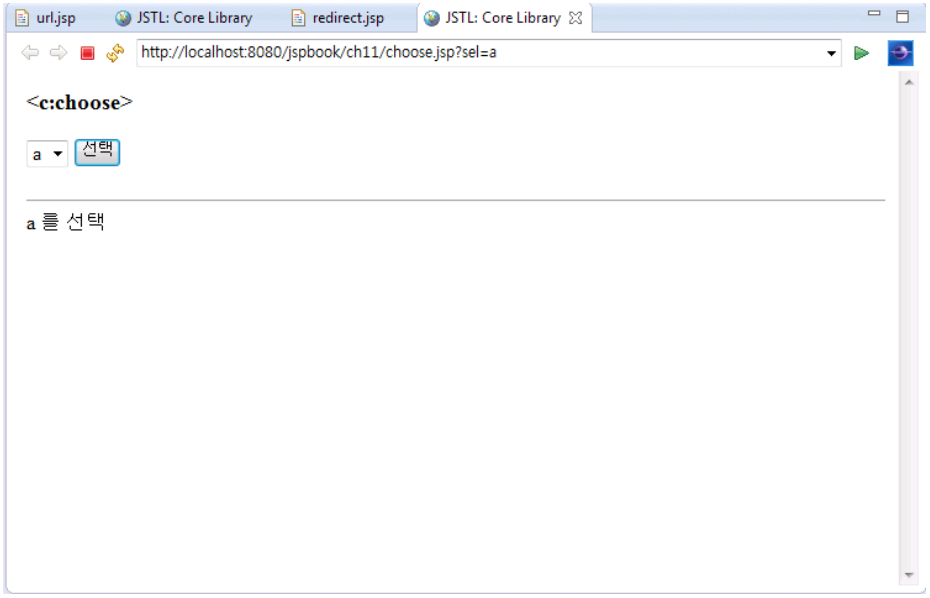
```
<c:redirect url="value" [context="context"]/>
<c:param> subtags
</c:redirect>
```

속성	필수	기본 값	설명
value	Y	없음	기본 URL
context	N	Current application	현재 웹 애플리케이션의 컨텍스트 이름

02. 핵심 라이브러리의 주요 태그

- redirect.jsp
  - 파라미터를 추가해 다른 페이지로 이동시킨다.
  - 앞에서 만들었던 choose.jsp에 선택 값을 파라미터로 지정해 이동시켜 결과를 확인한다.

```
10  <c:redirect url="/ch11/choose.jsp">
11  <c:param name="sel">a</c:param>
12  </c:redirect>
```



02. 핵심 라이브러리의 주요 태그

- <c:param> 태그
  - import, url, redirect 와 함께 사용된다.
  - url에 파라미터를 GET 방식으로 추가한다.
    - 바디가 없는 경우

```
<c:param name="name" value="value"/>
```

    - 바디 내용을 속성 값으로 사용하는 경우

```
<c:param name="name">
parameter value
</c:param>
```

속성	필수	기본 값	설명
url	Y	없음	현재 페이지 내에 포함시킬 URL
context	N	Current application	현재 웹 애플리케이션 컨텍스트 이름

CONTENTS

- JSTL의 종류
- 코어 태그
  - 변수 지원, 흐름 제어, URL 관련 태그, 기타 태그
- 국제화 태그
  - 로케일 지정
  - 메시지 처리 태그, 포맷팅

JSTL

- **JSP Standard Tag Library - 널리 사용되는 커스텀 태그를 표준으로 만든 태그 라이브러리 (자신만의 태그를 추가하는 기능 : Custom Tag)**
- **jsp 2.2와 호환되는 JSTL 버전은 1.2 (JSTL 1.2 버전은 JSP 2.1과도 호환)**
- **JSTL 태그 종류**

라이브러리	하위 기능	접두어	관련URI
코어	변수지원 흐름 제어 URL 처리	c	http://java.sun.com/jsp/jstl/core
XML	XML 코어 흐름 제어 XML 변환	x	http://java.sun.com/jsp/jstl/xml
국제화	지역 메시지 형식 숫자 및 날짜 형식	fmt	http://java.sun.com/jsp/jstl/fmt
데이터베이스	SQL	sql	http://java.sun.com/jsp/jstl/sql
함수	콜렉션 처리 String 처리	fn	http://java.sun.com/jsp/jstl/functions

■ JSTL 1.2 관련 jar 파일 필요

- 아래 사이트에서 jstl-1.2.jar 파일 다운로드
  - http://search.maven.org/#browse%7C-1829754406

- jstl-1.2.jar 파일을 WEB-INF/lib에 복사

코어 태그

기능분류	태그	설명
변수 지원	set	JSP에서 사용될 변수를 설정한다.
	remove	설정한 변수를 제거한다.
흐름 제어	if	조건에 따라 내부 코드를 수행한다.
	choose	다중 조건을 처리할 때 사용된다.
	forEach	콜렉션이나 Map의 각 항목을 처리할 때 사용된다.
	forTokens	구분자로 분리된 각각의 토큰을 처리할 때 사용된다.
URL 처리	import	URL을 사용하여 다른 자원의 결과를 삽입한다.
	redirect	지정한 경로로 리다이렉트 한다.
	url	URL을 재작성 한다.
기타 태그	catch	예외 처리에 사용된다.
	out	JspWriter에 내용을 알맞게 처리한 후 출력한다.

```
<% taglib prefix="c" url="http://java.sun.com/jsp/jstl/core"  %>
```

■ 변수 지원 태그

- 변수 설정
  - EL 변수 값 설정 (생성 또는 변경)
    - <c:set var="변수명" value="값" [scope="영역"] />
    - <c:set var="변수명" value="값" [scope="영역"]>값</c:set>
  - 특정 EL 변수의 프로퍼티 값 설정
    - <c:set target="대상" property="프로퍼티이름" value="값" />
    - <c:set target="대상" property="프로퍼티이름">값</c:set>
- 변수 삭제
  - <c:remove var="varName" [scope="영역"] />
    - scope 미지정시 모든 영역의 변수 삭제

■변수 지원 태그

use_c_set.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page import="mvjsp.chap16.Member" %>
<%@ page import="java.util.HashMap" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
        Member member = new Member();
        HashMap<String, String> pref = new HashMap<String, String>();
%>
<html>
<body>
<c:set var="member" value="<%= member %>" />
<c:set target="${member}" property="name" value="최범균" />

<c:set var="pref" value="<%= pref %>" />
<c:set var="favoriateColor" value="#{pref.color}" />
```

■ 변수 지원 태그

use_c_set.jsp

회원 이름: ${member.name},
좋아하는 색: ${favoriateColor}

<c:set target="${pref}" property="color" value="red" />

설정 이후 좋아하는 색: ${favoriateColor}

</body>
</html>

■ 흐름 제어

- **if - 조건이 true일 경우 몸체 내용 실행**

```
<c:if test="조건">
...
</c:if>
```

- **choose - when - otherwise**
  - **swich - case - default와 동일**

```
<c:choose>
  <c:when test="${member.level == 'trial'}" >
    ...
  </c:when>
  <c:when test="${member.level == 'regular'}" >
    ...
  </c:when>
  <c:otherwise>
    ...
  </c:otherwise>
</c:choose>
```

■ 흐름 제어

use_if_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>if 태그</title></head>
<body>
<c:if test="true">
무조건 수행<br>
</c:if>
<c:if test="${param.name == 'bk'}">
name 파라미터의 값이 ${param.name} 입니다.<br>
</c:if>

<c:if test="${18 < param.age}">
당신의 나이는 18세 이상입니다.
</c:if>
</body>
</html>
```

■ 흐름 제어

use_choose_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html><head><title>choose 태그</title></head><body>
<ul>
<c:choose>
  <c:when test="${param.name == 'bk'}" >
    <li> 당신의 이름은 ${param.name} 입니다.
  </c:when>
  <c:when test="${param.age >= 20}" >
    <li> 당신은 20세 이상입니다.
  </c:when>
  <c:otherwise>
    <li> 당신은 'bk'가 아니고 20세 이상이 아닙니다.
  </c:otherwise>
</c:choose>
</ul>
</body></html>
```

■ 반복 처리

- **forEach**
  - **집합이나 콜렉션 데이터 사용**

```
<c:forEach var="변수" items="아이템">
  … ${변수사용} ...
</c:forEach>
```

  - **특정 회수 반복**

```
<c:forEach var="i" begin="1" end="10" [step="값"]>
  ${i} 사용
</c:forEach>
```

■ 반복 처리

- **forEach**
  - **varStatus 속성**

```
<c:forEach var="item" items="<%= someItemList %>" varStatus="status">
${status.index + 1} 번째 항목 : ${item.name}
</c:forEach>
```

index - 루프 실행에서 현재 인덱스, count - 루프 실행 회수

begin - begin 속성 값, end - end 속성 값, step - step 속성 값

first - 현재 실행이 첫 번째 실행인 경우 true

last - 현재 실행이 루프의 마지막 실행인 경우 true

current - 콜렉션 중 현재 루프에서 사용할 객체

■ 반복 처리

use_foreach_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ page import="java.util.HashMap" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
HashMap<String,Object> mapData = new HashMap<String,Object>();
mapData.put("name", "최범균");
mapData.put("today", new java.util.Date());
%>
<c:set var="intArray" value="<%= new int[] {1,2,3,4,5} %>" />
<c:set var="map" value="<%= mapData %>" />
<html><head><title>forEach 태그</title></head><body>
<h4>1부터 100까지 홀수의 합</h4>
<c:set var="sum" value="0" />
<c:forEach var="i" begin="1" end="100" step="2">
<c:set var="sum" value="${sum + i}" />
</c:forEach>
결과 = ${sum}
```

■ 반복 처리

use_foreach_tag.jsp

```
<h4>구구단: 4단</h4>
<ul>
<c:forEach var="i" begin="1" end="9">
<li>4 * ${i} = ${4 * i}
</c:forEach>
</ul>
<h4>int형 배열</h4>
<c:forEach var="i" items="${intArray}" begin="2" end="4" varStatus="status">
        ${status.index}-${status.count}-[${i}] <br />
</c:forEach>
<h4>Map</h4>
<c:forEach var="i" items="${map}">
${i.key} = ${i.value}<br>
</c:forEach>
</body>
</html>
```

■ 반복 처리

use_fortokens_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head><title>forTokens 태그</title></head>
<body>

콤마와 점을 구분자로 사용:<br>
<c:forTokens var="token"
             items="빨강색,주황색.노란색.초록색,파랑색,남색.보라색"
             delims=",.">
${token}
</c:forTokens>

</body>
</html>
```

URL 관련 태그

- **import - 외부/내부 페이지를 현재 위치에 삽입**

```
<c:import url="URL" [var="변수명"] [scope="영역"] [charEncoding="캐릭터셋"]>
    <c:param name="파라미터이름" value="값" />
    ...
</c:import>
```

  - **상대 URL import 시 <jsp:include>와 동일하게 동작**

- **url - 절대 URL과 상대 URL을 알맞게 생성**

```
<c:url value="URL" [var="varName"] [scope="영역"]>
   <c:param name="이름" value="값" />
</c:url>
```

  - **웹 컨텍스트 내에서 절대 경로 사용시 컨텍스트 경로 자동 추가**

- **redirect - 지정한 페이지로 리다이렉트**

```
<c:redirect url="URL" [context="콘텍스트경로"]>
    <c:param name="이름" value="값" />
</c:redirect>
```

URL 관련 태그

use_import_tag.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<% request.setCharacterEncoding("utf-8"); %>
<c:choose>
        <c:when test="${param.type == 'flickr'}">
                <c:import url="http://www.flickr.com/search/">
                        <c:param name="f" value="hp" />
                        <c:param name="q" value="보라매공원" />
                </c:import>
        </c:when>
        <c:when test="${param.type == 'youtube'}">
                <c:import url="http://www.youtube.com/results">
                        <c:param name="search_query" value="보라매 공원" />
                </c:import>
        </c:when>
```

URL 관련 태그

use_import_tag.jsp

```
        <c:otherwise>
            <c:import url="use_import_tag_help.jsp">
                <c:param name="message" value="선택해주세요" />
            </c:import>
        </c:otherwise>
</c:choose>
```

URL 관련 태그

use_import_tag_help.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>도움</title></head>
<body>
${param.message}:
<ul>
        <li>flickr - Flickr에서 검색</li>
        <li>youtube - 유튜브에서 검색</li>
</ul>
</body>
</html>
```

URL 관련 태그

use_url_tag.jsp

```
<%@ page contentType="text/html; charset=euc-kr" session="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
<c:url value="http://search.daum.net/search" var="searchUrl">
        <c:param name="w" value="blog" />
        <c:param name="q" value="공원" />
</c:url>

<ul>
        <li>${searchUrl}</li>
        <li><c:url value="/use_if_tag.jsp" /></li>
        <li><c:url value="./use_if_tag.jsp" /></li>
</ul>

</body>
</html>
```

■ 기타 코어 태그

- **out - 데이터를 출력**

```
<c:out value="value" [escapeXml="(true|false)"] [default="defaultValue"] />

<c:out value="value" [escapeXml="(true|false)"]>
    default value
</c:out>
```

  - **escapeXml 속성이 true일 경우 다음과 같이 특수 문자 처리**
    - **< → < , > → >**
    - **& → & , ' → ' , " → "**

- **catch - 몸체에서 발생한 예외를 변수에 저장**

```
<c:catch var="exName">
...
예외가 발생할 수 있는 코드
...
</c:catch>
${exName} 사용
```

■ 기타 코어 태그

use_out_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ page import = "java.io.IOException, java.io.FileReader" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>소스 보기</title></head>
<body>

<%
        FileReader reader = null;
        try {
                String path = request.getParameter("path");
                reader = new FileReader(getServletContext().getRealPath(path));
%>
<pre>
소스 코드 = <%= path %>
```

■ 기타 코어 태그

use_out_tag.jsp

```
<c:out value="<%= reader %>" escapeXml="true" />
</pre>
<%
        } catch(IOException ex) {
%>
에러: <%= ex.getMessage() %>
<%
        } finally {
                if (reader != null)
                        try {
                                reader.close();
                        } catch(IOException ex) {}
        }
%>

</body>
</html>
```

■ 기타 코어 태그

use_catch_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>catch 태그</title></head>
<body>
<c:catch var="ex">
name 파라미터의 값 = <%= request.getParameter("name") %><br>
<% if (request.getParameter("name").equals("test")) { %>
${param.name}은 test 입니다.
<% } %>
</c:catch>
<p>
<c:if test="${ex != null}">
익셉션이 발생하였습니다:<br>
${ex}
</c:if>
</body></html>
```

국제화 태그

기능분류	태그	설명
로케일 지정	setLocale	Locale을 지정한다.
	requestEncoding	요청 파라미터의 캐릭터 인코딩을 지정
메시지 처리	bundle	사용할 번들을 지정
	message	지역에 알맞은 메시지를 출력
	setBundle	리소스 번들을 읽어와 특정 변수에 저장
숫자 및 날짜 포맷팅	formatNumber	숫자를 포맷팅
java.util.date 사용 →	formatDate	Date 객체를 포맷팅
	parseDate	문자열로 표시된 날짜를 분석해서 Date 객체로 변환
	parseNumber	문자열로 표시된 날짜를 분석해서 숫자로 변환
	setTimeZone	시간대 정보를 특정 변수에 저장
	timeZone	시간대를 지정

■ 로케일 지정 및 요청 파라미터 인코딩 지정

- <fmt:setLocale value="언어코드" scope="범위" />
  - 국제화 태그가 Accept-Language 헤더에서 지정한 언어가 아닌 다른 언어를 사용하도록 지정하는 기능

- <fmt:requestEncoding value="캐릭터셋" />
  - 요청 파라미터의 캐릭터 인코딩을 지정
  - request.setCharacterEncoding("캐릭터셋")과 동일

■ 국제화 태그

message.properties　　(이클립스의 경우는 src/resource)

TITLE = MadVirus's Learning JSP 2.1

GREETING = HI! I'm BK

VISITOR = Your ID is **{0}.**

■ 국제화 태그

message_ko.properties.src　　(이클립스의 경우는 src/resource)

TITLE = 최범균의 JSP 2.0 배우기
GREETING = 안녕하세요. 최범균입니다.
VISITOR = 당신의 아이디는 {0}입니다.

■ 국제화 태그

message_ko.properties　　(이클립스의 경우는 src/resource)

TITLE = ₩ucd5c₩ubc94₩uade0₩uc758 JSP 2.1 ₩ubc30₩uc6b0₩uae30

GREETING = ₩uc548₩ub155₩ud558₩uc138₩uc694.
₩ucd5c₩ubc94₩uade0₩uc785₩ub2c8₩ub2e4.

VISITOR = ₩ub2f9₩uc2e0₩uc758 ₩uc544₩uc774₩ub514₩ub294
{0}₩uc785₩ub2c8₩ub2e4.

<fmt:message> 태그

- 리소스 번들 범위에서 메시지 읽기

```
<fmt:bundle basename="resource.message" [prefix="접두어"]>
   <fmt:message key="GREETING" />
</fmt:bundle>
```

- 지정한 번들에서 메시지 읽기

```
<fmt:setBundle var="message" basename="resource.message" />
...
<fmt:message bundle="${message}" key="GREETING" />
```

- <fmt:message> 태그의 메시지 읽는 순서
  - bundle 속성에 지정한 리소스 번들을 사용
  - <fmt:bundle> 태그에 중첩된 경우 <fmt:bundle> 태그에서 설정한 리소스 번들 사용
  - 1과 2가 아닐 경우 기본 리소스 번들 사용. 기본 리소스 번들은 web.xml 파일에서 javax.servlet.jsp.jstl.fmt.localizationContext 콘텍스트 속성을 통해서 설정 가능

■ <fmt:message> 태그

use_message_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- <fmt:setLocale value="en" /> --%>
<fmt:bundle basename="resource.message">
<fmt:message key="TITLE" var="title"/>
<html><head><title>${title}</title></head><body>
<fmt:message key="GREETING" />
<br>
<c:if test="${! empty param.id}">
<fmt:message key="VISITOR">
<fmt:param value="${param.id}" />
</fmt:message>
</c:if>
</body>
</html>
</fmt:bundle>
```

■ <fmt:message> 태그

use_message_tag2.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<fmt:setBundle var="message" basename="resource.message" />
<fmt:message bundle="${message}" key="TITLE" var="title"/>
<html>
<head><title>${title}</title></head>
<body>
<fmt:message bundle="${message}" key="GREETING" />
<br>
<c:if test="${! empty param.id}">
<fmt:message bundle="${message}" key="VISITOR">
    <fmt:param value="${param.id}" />
</fmt:message>
</c:if>
</body>
</html>
```

formatNumber 태그

- **숫자를 포맷팅**

```
<fmt:formatNumber value="숫자값" [type="값타입"] [pattern="패턴"]
    [currentCode="통화코드"] [currencySymbol="통화심볼"]
    [groupingUsed="(true|false)"] [var="변수명"] [scope="영역"] />
```

- **주요 속성**

속성	표현식/EL	타입	설명
value	사용 가능	String 또는 Number	양식에 맞춰 출력할 숫자
type	사용 가능	String	어떤 양식으로 출력할지를 정한다. number는 숫자형식, percent는 % 형식, currency는 통화형식으로 출력. 기본 값은 number.
pattern	사용 가능	String	직접 숫자가 출력되는 양식을 지정한다. DecimalFormat 클래스에서 정의되어 있는 패턴 사용
var	사용 불가	String	포맷팅 한 결과를 저장할 변수 명. var 속성을 사용하지 않으면 결과가 곧바로 출력.
scope	사용 불가	String	변수를 저장할 영역. 기본 값은 page 이다.

formatNumber 태그

use_number_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>numberFormat 태그 사용</title></head>
<body>
<c:set var="price" value="10000" />
<fmt:formatNumber value="${price}" type="number" var="numberType" />
<br/>
통화: <fmt:formatNumber value="${price}" type="currency" currencySymbol="원" />
<br/>
퍼센트: <fmt:formatNumber value="${price}"  type="percent" groupingUsed="false" />
<br/>
숫자: ${numberType}
<br/>
패턴: <fmt:formatNumber value="${price}" pattern="00000000.00"/>
</body>
</html>
```

parseNumber 태그

- **문자열을 숫자 데이터 타입으로 변환**

```
<fmt:parseNumber value="값" [type="값타입"] [pattern="패턴"]
    [parseLocale="통화코드"] [integerOnly="true|false"]
    [var="변수명"] [scope="영역"] />
```

- **주요 속성**

속성	표현식/EL	타입	설명
value	사용 가능	String	파싱할 문자열
type	사용 가능	String	value 속성의 문자열 타입을 지정. number, currency, percentage 가 올 수 있다. 기본 값은 number
pattern	사용 가능	String	직접 파싱할 때 사용할 양식을 지정
var	사용 불가	String	파싱한 결과를 저장할 변수 명을 지정
scope	사용 불가	String	변수를 저장할 영역을 지정한다. 기본 값은 page.

formatDate 태그

- **날짜 정보를 담은 객체(Date)를 포맷팅**

```
<fmt:formatDate value="날짜값"
    [type="타입"] [dateStyle="날짜스타일"] [timeStyle="시간스타일"]
    [pattern="패턴"] [timeZone="타임존"]
    [var="변수명"] [scope="영역"] />
```

- **주요 속성**

속성	표현식/EL	타입	설명
value	사용 가능	java.util.Date	포맷팅할 날짜 및 시간 값
type	사용 가능	String	날짜, 시간 또는 둘 다 포맷팅 할 지의 여부를 지정
dateStyle	사용 가능	String	날짜에 대한 포맷팅 스타일을 지정
timeStyle	사용 가능	String	시간에 대한 포맷팅 스타일을 지정
pattern	사용 가능	String	직접 파싱할 때 사용할 양식을 지정
var	사용 불가	String	파싱한 결과를 저장할 변수 명을 지정
scope	사용 불가	String	변수를 저장할 영역을 지정

formatDate 태그

use_date_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>dateFormat태그 사용</title></head>
<body>

<c:set var="now" value="<%= new java.util.Date() %>" />
<fmt:formatDate value="${now}" type="date" dateStyle="full" /> <br>
<fmt:formatDate value="${now}" type="date" dateStyle="short" /> <br>
<fmt:formatDate value="${now}" type="time" /> <br>
<fmt:formatDate value="${now}" type="both"
                dateStyle="full" timeStyle="full" /> <br>
<fmt:formatDate value="${now}" pattern="z a h:mm" /> <br>

</body>
</html>
```

timeZone과 setTimeZone

- **국제화 태그가 사용할 시간대 설정**

```
<fmt:timeZone value="Hongkong">
  <!-- 사용하는 시간을 Hongkong 시간대에 맞춘다. -->
  <fmt:formatDate ... />
</fmt:timeZone>
```

timeZone과 setTimeZone

use_timezone_tag.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>timeZone 태그 사용</title></head>
<body>
<c:set var="now" value="<%= new java.util.Date() %>" />
<fmt:formatDate value="${now}" type="both"
                dateStyle="full" timeStyle="full" />
<br>
<fmt:timeZone value="Hongkong">
<fmt:formatDate value="${now}" type="both"
                dateStyle="full" timeStyle="full" />
</fmt:timeZone>

</body>
</html>
```

timeZone과 setTimeZone

listTimezones.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>시간대 목록</title></head>
<body>

<c:forEach var="id" items="<%= java.util.TimeZone.getAvailableIDs() %>">
    ${id}<br/>
</c:forEach>

</body>
</html>
```

web.xml, 국제화 태그 콘텍스트 속성

속성 이름	설명
javax.servlet.jsp.jstl.fmt.localizationContext	기본으로 사용할 리소드 번들을 지정한다. 리소스 번들의 basename을 입력한다.
javax.servlet.jsp.jstl.fmt.locale	기본으로 사용할 로케일을 지정한다.
javax.servlet.jsp.jstl.fmt.timeZone	기본으로 사용할 시간대를 지정한다.

JSTL이 제공하는 주요 EL 함수

함수	설명
length(obj)	obj가 List와 같은 Collection인 경우 저장된 항목의 개수를 리턴하고, obj가 문자열일 경우 문자열의 길이를 리턴한다.
toUpperCase(str)	str을 대문자로 변환한다.
toLowerCase(str)	str을 소문자로 변환한다.
substring(str, idx1, idx2)	str.substring(idx1, idx2)의 결과를 리턴한다. idx2가 -1일 경우 str.substring(idx1)과 동일하다.
trim(str)	str 좌우의 공백문자를 제거한다.
replace(str, src, dest)	str에 있는 src를 dest로 변환한다.
indexOf(str1, str2)	str1에서 str2가 위치한 인덱스를 구한다.
startsWith(str1, str2)	str1이 str2로 시작할 경우 true를, 그렇지 않을 경우 false를 리턴한다.
endsWith(str1, str2)	str1이 str2로 끝나는 경우 true를, 그렇지 않을 경우 false를 리턴한다.
contains(str1, str2)	str1이 str2를 포함하고 있을 경우 true를 리턴한다.
escapeXml(str)	XML의 객체 참조에 해당하는 특수 문자를 처리한다. 예를 들어, '&'는 '&'로 변환한다.

■ JSTL이 제공하는 주요 EL 함수

use_function.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<html>
<head><title>함수 사용</title></head>
<body>
<c:set var="str1" value="Functions <태그>를 사용합니다. " />
<c:set var="str2" value="사용" />
<c:set var="tokens" value="1,2,3,4,5,6,7,8,9,10" />

length(str1) = ${fn:length(str1)} <br>
toUpperCase(str1) = "${fn:toUpperCase(str1)}" <br>
toLowerCase(str1) = "${fn:toLowerCase(str1)}" <br>
substring(str1, 3, 6) = "${fn:substring(str1, 3, 6)}" <br>
substringAfter(str1, str2) = "${fn:substringAfter(str1, str2)}" <br>
substringBefore(str1, str2) = "${fn:substringBefore(str1, str2)}" <br>
```

■ JSTL이 제공하는 주요 EL 함수

use_function.jsp

```
trim(str1) = "${fn:trim(str1)}" <br>
replace(str1, src, dest) = "${fn:replace(str1, " ", "-")}" <br>
indexOf(str1, str2) = "${fn:indexOf(str1, str2)}" <br>
startsWith(str1, str2) = "${fn:startsWith(str1, 'Fun')}" <br>
endsWith(str1, str2) = "${fn:endsWith(str1, "합니다.")}" <br>
contains(str1, str2) = "${fn:contains(str1, str2)}" <br>
containsIgnoreCase(str1, str2) = "${fn:containsIgnoreCase(str1, str2)}" <br>

<c:set var="array" value="${fn:split(tokens, ',')}" />

join(array, "-") = "${fn:join(array, "-")}" <br>
escapeXml(str1) = "${fn:escapeXml(str1)}" <br>

</body>
</html>
```